Beamline Micro-P

Enhancements for fast magnet supply

Mon, Jan 14, 2002

This note is the result of communications with Bob Florian and Arlene Lennox about changes to be made to the Beamline micro-p software to optimize performance in switching between NTF beam cycles and HEP beam cycles. The purpose for this is to share beam cycles in the most efficient manner, taking advantage of recent hardware optimizations in the 58 degree bending magnet power supply that were made to reduce the time of ramping the magnet that directs Linac beam into the NTF beam line.

The reading of the magnet current is actually filtered from the real signal, in order to take away some noise in that current signal. Without the filter, the noise is bad enough that it can make the reading of the current appear to be outside the tolerance limits. The magnet current does not really exhibit this 720 Hz noise, but the magnet current transducer output signal that arrives at the A/D does. But the RC filtering introduces a delay, so that the Beamline micro-p is slightly behind in seeing the real current value. And that means it may not be aware that the current has really arrived at its intended nominal level, even though it really has.

Bob Florian plotted some of the readings of interest on the usual kind of 15 Hz plot available in the control system. While this may not show the detail that Milorad Popovic captured via scope traces, it does show the data on which the Beamline micro-p makes its decisions. He tested the case where a single HEP beam pulse was scheduled during NTF operation. Indeed, when the magnet switched off, the very next cycle exhibited HEP beam. But when the magnet switched back on, the reading of its current was a few percent shy of the nominal value; of course, there was no NTF beam on that cycle. On the following cycle, the reading showed that the magnet was up to nominal, but there was still no NTF beam. Altogether, there were 3 NTF beam pulses lost to grant one HEP beam cycle; there were 2 cycles of lost opportunities for NTF beam.

Milorad's data apparently showed that the magnet current was really up to nominal within one 15 Hz cycle. But our reading may be lagging so that we don't know it yet. If we knew the truth, and we read it soon enough, that should eliminate one lost cycle.

When the `MPENBL` (micro-p enable) routine, which is invoked as the last significant piece of logic during 15 Hz processing, notices that the ramp enable status just went high, compared with being low on the previous cycle, it checks the number of delay pulses specified as a minimum for waiting for the magnet to ramp up. If that value is zero, it thinks that someone forgot to set it, and it uses 2 by default; otherwise, it takes the value found there, which could be 1 as a minimum. (Recall the that original implementation of the power supply had the ramp requiring maybe 5 or more cycles; furthermore, it exhibited overshoot and some ringing.) Then it initializes the `ON/OFF` sequencing logic to the `OFF` state, using that counter value. Although the rest of the `MPENBL` logic might result in clearing the micro-p enable control line, nothing in that routine sets it; rather, the time it gets set is about midway through the 15 Hz cycle, when it knows that the next pulse is scheduled to be an `ON` pulse. (Recall the notion of duty cycle control via a sequence of `ON` pulses followed by a sequence of `OFF` pulses.) It knows about this time of midway through the cycle, because it actually arranges to get 60 Hz interrupts within each 15 Hz cycle by operating a timer.

In order to schedule an `ON` cycle immediately following a cycle on which the ramp enable status just went high, the logic must be changed to accept a value of zero for the number of

delay pulses for awaiting the magnets to ramp. (The current logic thinks that zero is an error, so it uses a 2-cycle delay in that case.) If this delay is zero, it must switch to the ON pulse state and set the count for the scheduled number of ON pulses in a sequence—assuming that this count is nonzero, of course.

It is useful to know the exact timing of what the micro-p does during 15 Hz processing. The timing of its interrupt, derived from the accelerator timing system, is set to 1251 microseconds after the reset clock event time. This should place it well before beam time, which occurs about 2000 microseconds after event time. (The reason it operates ahead of beam time is so it can capture the "zero data" pedestals for the QP, X1, and X2 signals.) Then it reads the pressure, temperature, and digital input mux. It then waits for 800 microseconds (according to a source code comment) in order to allow the beam pulse to occur and complete. After this delay, it reads the 16-channel A/D (including QP), the 8-channel differential A/D (including X1 and X2), and the "safety system status," which includes the interlock chassis status signals. It then computes the ion chamber deltas, making use of the zero data values captured earlier. Next it checks for the interlock conditions, assuming that it has treatment enable status. These conditions detect missing beam pulses; and if this is the last ON pulse in the sequence of ON pulses, it also checks dose thresholds, ratios and voltages. There are LEDs associated with each of these potential bad conditions, and if any of those LEDs is set, it pulls down a control line to stop treatment. The next step compares the readings of bend magnet currents against the nominal values and updates the currents nominal control line accordingly. Finally, the MPENBL routine is invoked that is described above. That ends 15 Hz processing; the time required for all this is 2.7 ms, from diagnostics that are included in the NTF data pool, with the MPENBL routine running near the end of this time. The time from the start of the 15 Hz interrupt code to the time of having just digitized the 16-channel A/D is 2.0 ms.

The recent improvements in the response of the magnet power supply have made it possible to bring the magnet currents up to nominal within one 66 ms cycle. At this time, it appears that this ramp is accomplished in 57 ms, beginning immediately after the beam pulse of the previous cycle. (This can be stated as 59 ms past the Booster reset clock event, since Linac beam pulses occur at 2 ms after such a clock event.) But in order for the micro-p to verify that the current is actually at nominal and thus set the currents nominal control line, it must have a chance to read the currents signal very late in the cycle. Its 15 Hz activity now starts at 1.251 ms after event time, so it can collect “just-in-time” zero data. But this is too late for checking the reading of the magnet currents, because the interlock box must ok beam within 50 microseconds following event time, or no beam can be accelerated. This means we need to check the magnet currents reading a bit before the event time, as it would not be possible to acquire the data and check it within 50 microseconds.

It turns out that this may be possible to arrange. The micro-p actually schedules 60 Hz interrupt activity for itself within the 15 Hz cycle. One reason for doing this was to capture zero data one 60 Hz period before the beam pulse--in order to remove the effects of 60 Hz ripple noise on those signals. But we now read zero data just before the beam pulse, as described above. When the third 60 Hz interrupt is scheduled, during the second 60 Hz interrupt, we can change the timer delay value so it occurs later, say at 60 ms or so past event time. That interrupt activity can be changed to digitize the magnet current signals, check them, and thereby establish the correct level for the currents nominal control line. By doing this, it should be possible to accelerate Linac beam for NTF on the very next cycle following an HEP beam cycle.

Detailed changes in Beamline micro-p software

Module `MPEnbl.a`

In the `MPENBL` routine, if `CST+0`, a byte whose value specifies the delay to allow the magnet to ramp up before permitting beam, is less than 40, interpret it as before: assume that it represents a number of cycles delay before permitting beam, with at least one cycle guaranteed. If it has a value in the range 40–64, assume it specifies the number of milliseconds delay to wait in the current cycle before checking to see whether the magnet is up and within tolerance. Operationally, it means that when it is desired to operate with minimal delay, the time to examine the bend field current must be specified. And that time should not be more than 64 ms, as measured from the 15 Hz interrupt time that comes 1251 microsec after reset event time, in order to allow for time to finish the job before the start of the next cycle. (15 Hz timing must allow for variations in power line frequency, which is why 65 ms is not permitted.) We could write the code so that a value of 0 is interpreted as 64 ms, say. This may be more user-friendly.

Module `Acc.a`

In the `SIXTY` (60 Hz interrupt) routine, on the occasion of the second 60 Hz interrupt, check the value of the `CST+0` byte. If it is at least 40, but not more than 64, compute the appropriate delay in 800 KHz units for the timer to produce the third 60 Hz interrupt. This arithmetic should be (`N*800 - 26666`), where `N` is the value of the `CST+0` byte. This computation would yield results in the range 5334-24534. (If the value of the `CST+0` byte is outside the expected range, do the same thing as now, and set 15000 into the timer register, resulting in a third interrupt time of one 60 Hz cycle before beam time.)

For the third 60 Hz interrupt, which will likely be later than that as a result of the new logic, call `RDAD` and `COMP` to read the 16-channel A/D and perform the checks of the two magnet readings against their nominal and tolerance values, setting the bend currents nominal control line appriopriately. We can remove the code that establishes zero-data values for `X1` and `X2` from this code, since this logic is also performed just before beam time.

The `CLOCK` routine can be called to get the time when the third "60 Hz" interrupt routine completes. Such time values are in units of 320 us. There are 8 bytes of such times that can be observed in the readings of `node061c` channels `013e`, `013f`, `0140`, `0141`, where each channel reading includes a pair of byte-wide time values. For this new time that marks the completion of the third 60 Hz interrupt, use `TIMES+7`, the last of the 8 bytes. This will leave one byte unused, `TIMES+6`. The table of `TIMES` bytes are as follows:

TIMES+	*Usual*	*Time, ms*	*Meaning*
0	`D0`	66.6	Length of previous 15 Hz cycle.
1	`06`	1.9	In 15 Hz interrupt, just read 16-channel A/D.
2	`08`	2.6	In 15 Hz interrupt, all data ready in data pool.
3	`08`	2.6	End of 15 Hz interrupt. All data in shared memory.
4	`68`	33.3	Start of second 60 Hz interrupt.
5	`7D-C8`	40.0-64.0	Start of third 60 Hz interrupt. (depends on `CST+0`)
6	`--`		spare
7	`7E-C9`	40.3-64.3	End of third 60 Hz interrupt.

To relate the above times to a delay after the reset event, add 1.25 ms. Also, the resolution of these measured one-byte time values is 0.32 ms, so the accuracy of the values here represented to 0.1 ms is somewhat optimistic.